National Park Service

Inventory and Monitoring Division

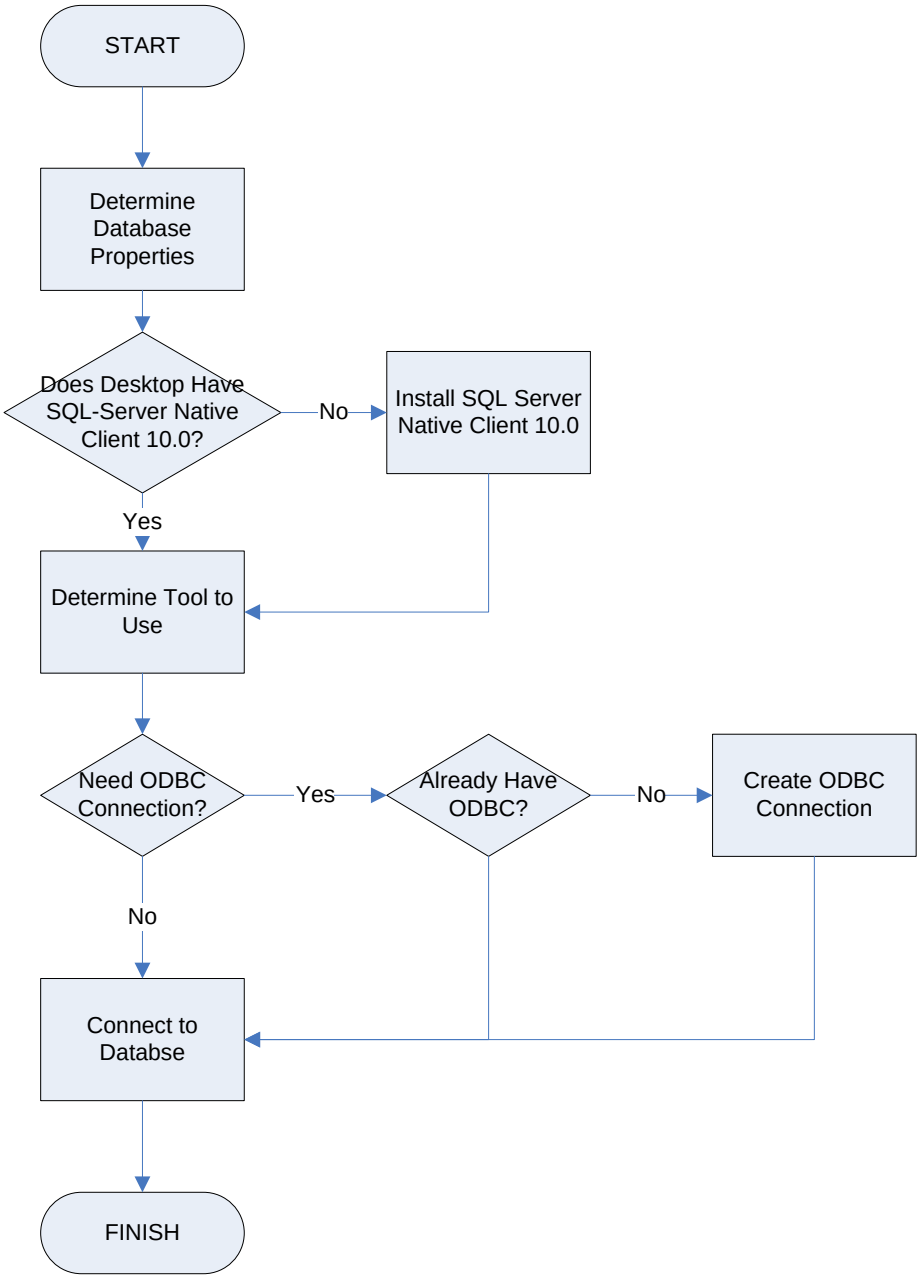
How to Connect Remotely to a SQL-Server Database

Revision Date:20 June 2012
Author: Brent Frakes; Simon Kingston, Tom Philippi

Table of Contents

1. Introduction

This SOP documents the process for connecting remotely to a SQL-Server database. The following figure shows the basic steps which are further detailed in the following sections.



2. Determine Database Properties

To connect to a SQL-Server database, you will need to know the following:

- Server Name – The path to the server. In most cases, it will look something like the following: INP2300FCSGUMO6\GIS_Working_Data
- Database Name – The name of the database on the server.

SQL Server supports two types of authentication, so you will need to determine which one to use:

- Windows Authentication - You are automatically authenticated by logging on to your computer.
- SQL-Server Authentication – You will need to obtain a UserName and Password from the database administrator.

3. SQL Server Native Client 10.0

If your computer does not already have the 10.0 Client, you will need to download it from Microsoft. In general, the native client should be on most NPS computers. When in doubt, the following path gives you directions:

http://msdn.microsoft.com/en-us/data/aa937733.aspx

4. Determine Application and Connection Type to Use

It is possible to connect to SQL-Server using a variety of applications. Some can connect natively, while others require an open database connection (ODBC). The type of connection needs to be established before it can connect to SQL-Server. Below is a table of the most common applications with an indication of the supported and recommended connection types.

Application	Connection	Notes
SQL Server Management Studio	Native	This is the best way to connect to any SQL-Server database. It has fully querying capabilities.
MS Access	Both	Ideal if developing customized forms and reports. Unfortunately, there are some timeout issues with larger queries.
MS Excel	Both	Ideal for creating dynamic tables and graphs
MS Word	Both	Ideal for dynamically embedding graphs and tables in reports

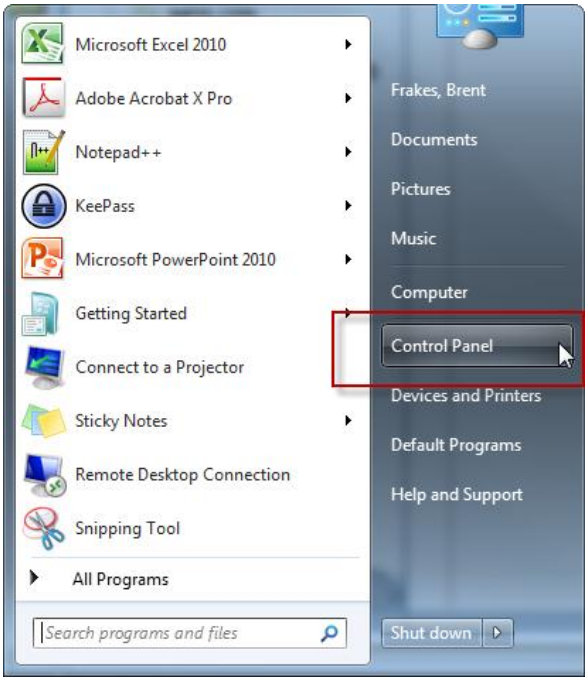
Python	Both. ODBC recommended for Windows Authentication.	Allows customized coding with python language
ArcGIS	ODBC	Integrate tables into mapping applications
R	Both. ODBC recommended for Windows Authentication.	Open source statistical software

If the application you intend to use requires ODBC, refer to Section 5. Otherwise, proceed to the section for your software.

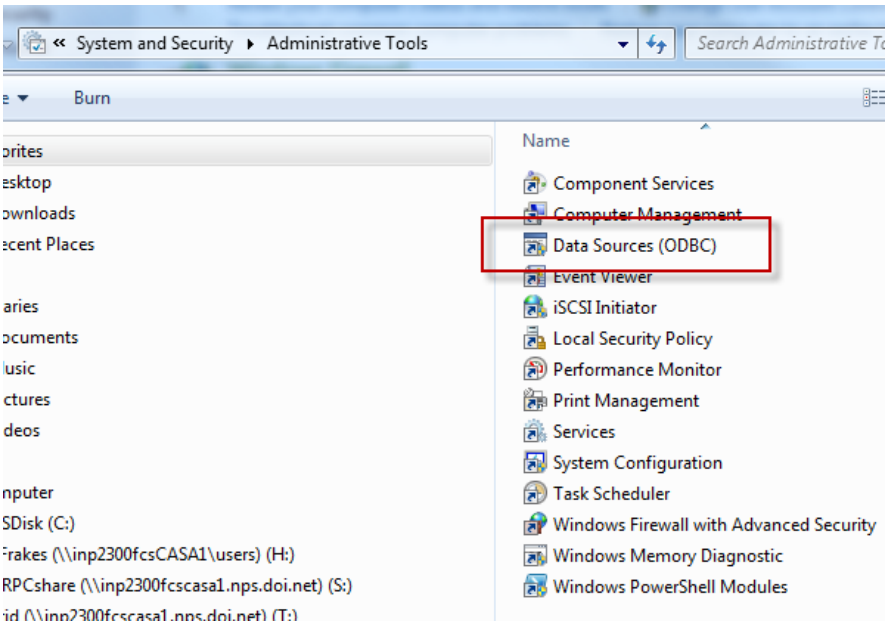
5. Establishing an Open Database Connection (ODBC)

The following section shows you how to establish an ODBC connection to SQL-Server.

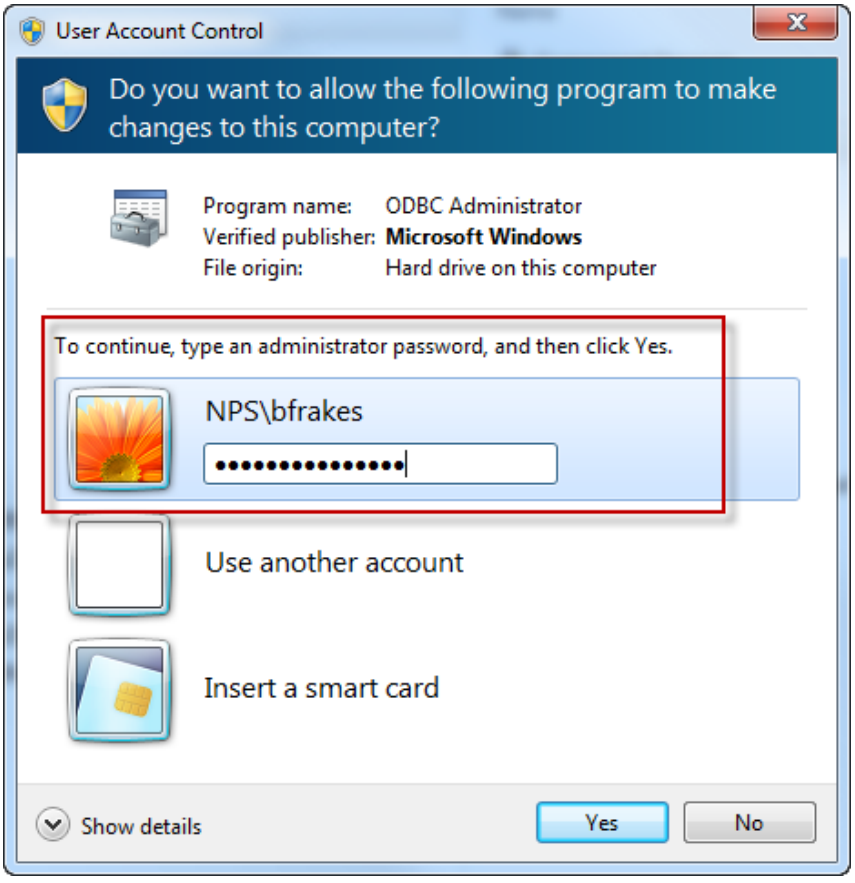
1. Be sure you have SQL Server Native Client 10.0. If you don't, see Section 3.
2. Select Start → Control Panel



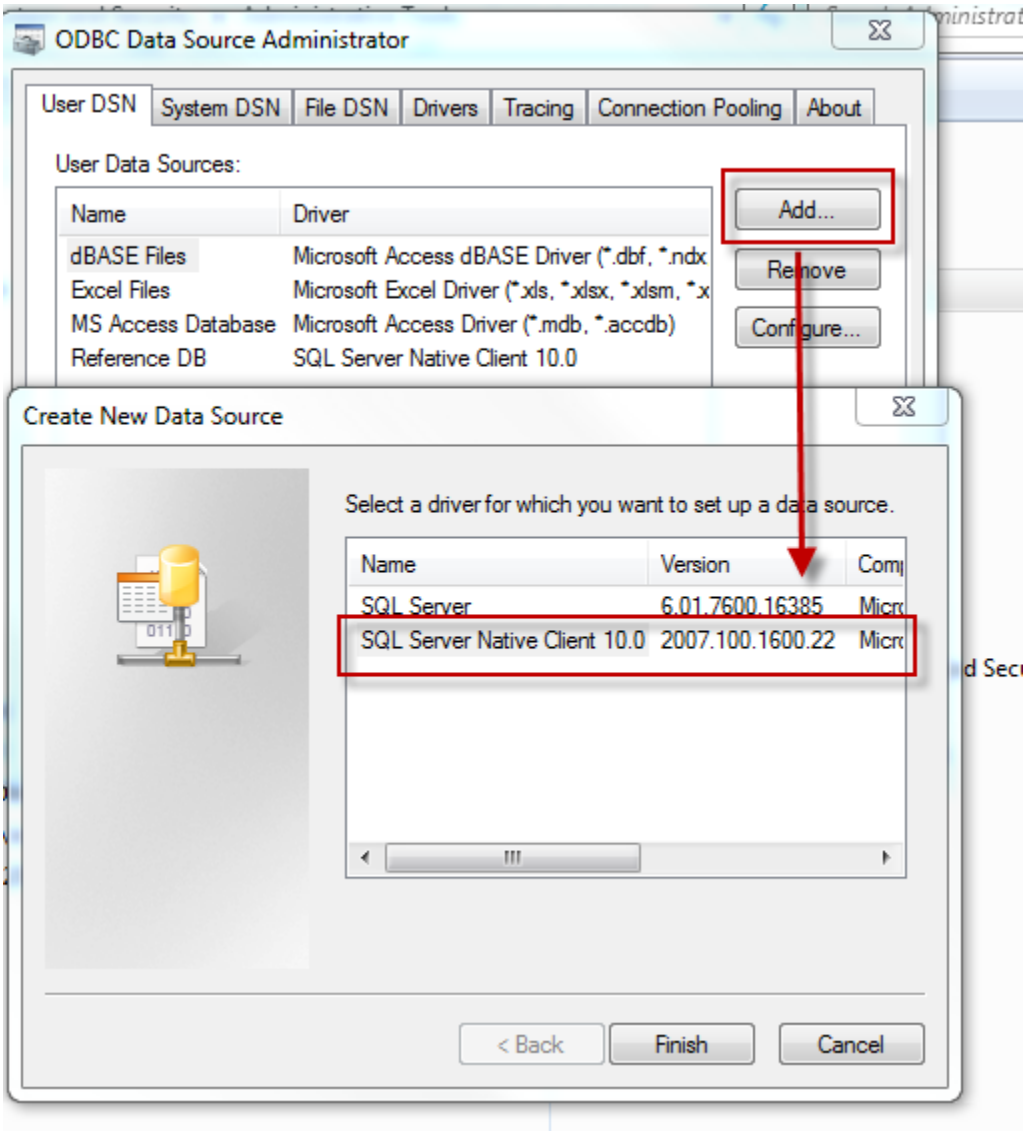
3. Select System and Security→Administrative Tools → Data Source(ODBC)



4. If prompted for an Administrator password, re-type in your password. Do not use any local installer account.



5. Select the 'User[1] DSN Tab' and click on the Add Button

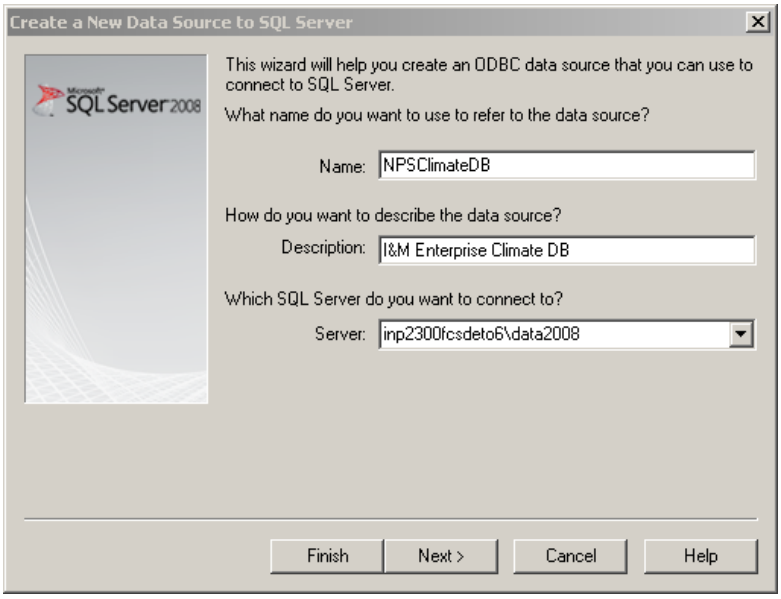


6. Choose the SQL Server Native Client 10.0

[1] If you have administrative permissions, you will also be able to set up System DSNs.

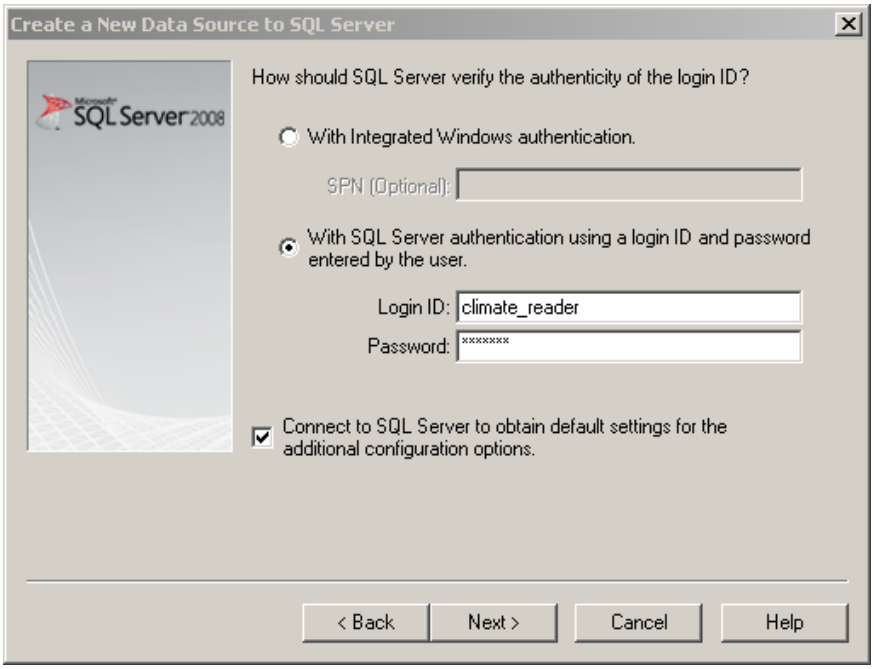
7. Specify the following:

- Name – Give the database connection an intuitive name
- Description – Give the database an informative description
- Server – Provide the Server Name

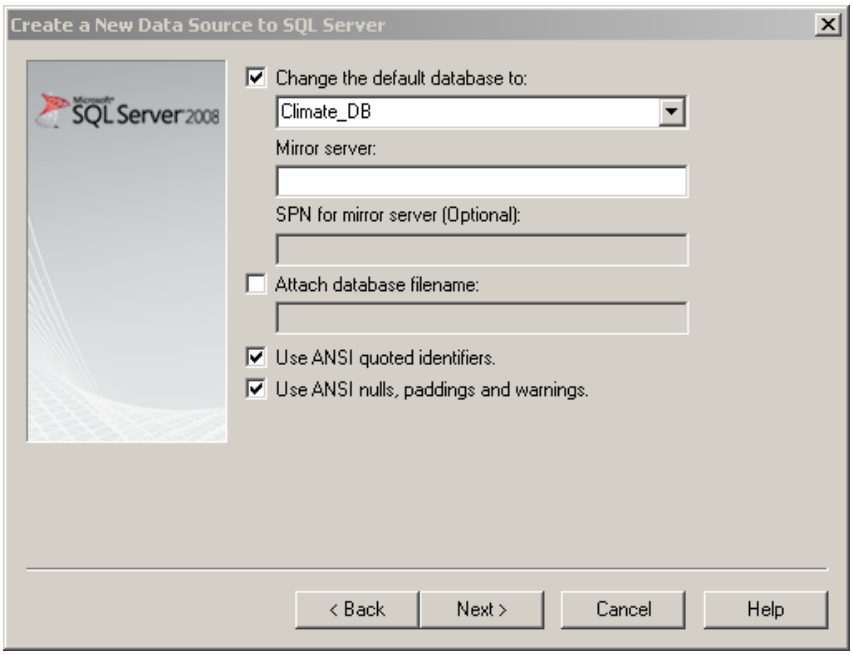


8. Select the appropriate Authentication Option

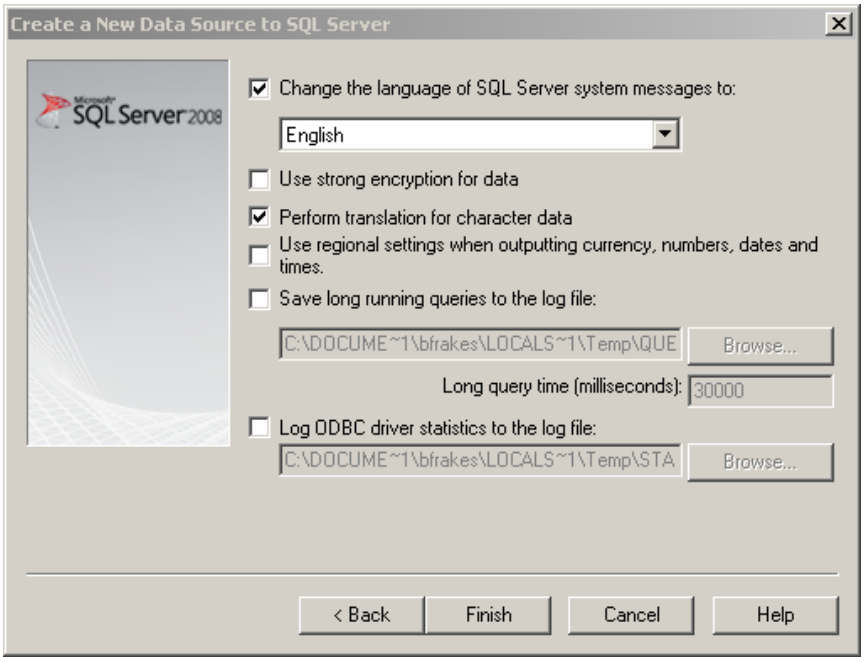
- Integrated Windows Authentication – Uses your active directory user name and password
- SQL Server Authentication – Uses a specied username and password



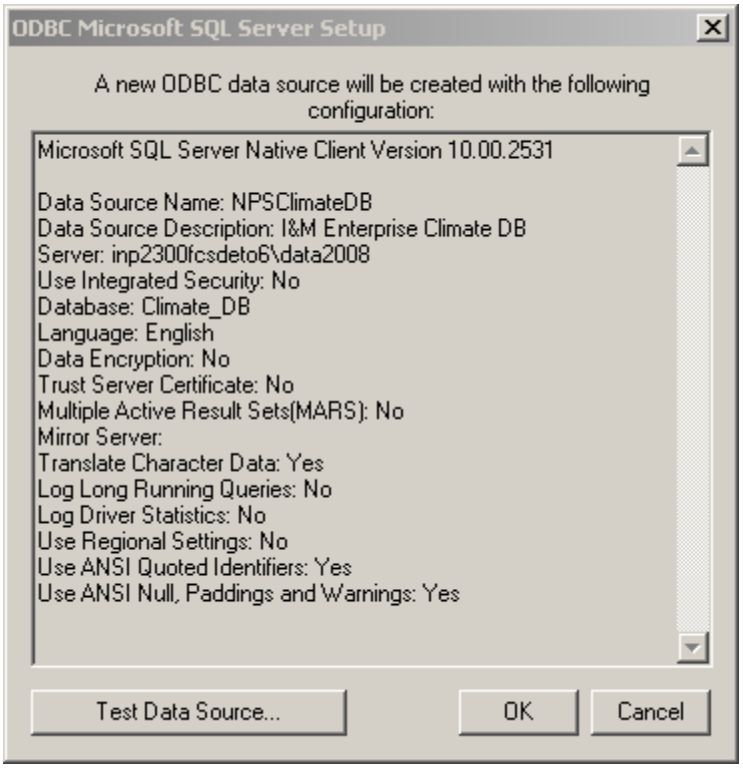
9. Select the Change Default Database



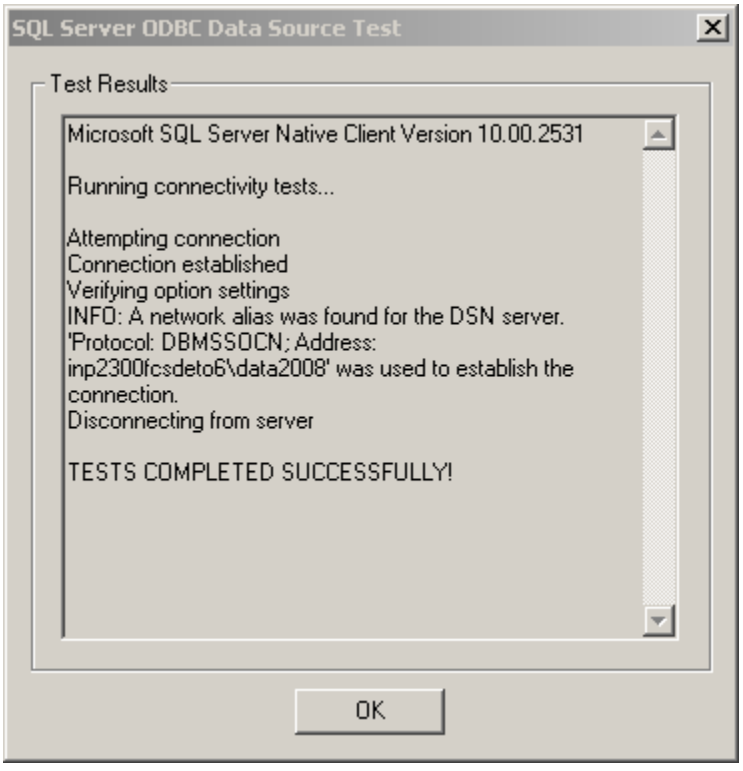
10. Make sure the language is set to English



11. Select Finish



12. Test Data Source



If you see ‘TESTS COMPLETED SUCCESSFULLY!’, then you are good to go.

6. Connect To SQL-Server Database

Below are directions for connecting to the SQL Database using a variety of applications.

6.1. SQL-Server 2008 Management Studio Express (SSME)

1. Download and install the SQL-Server 2008 Management Studio Express application appropriate for your computer.

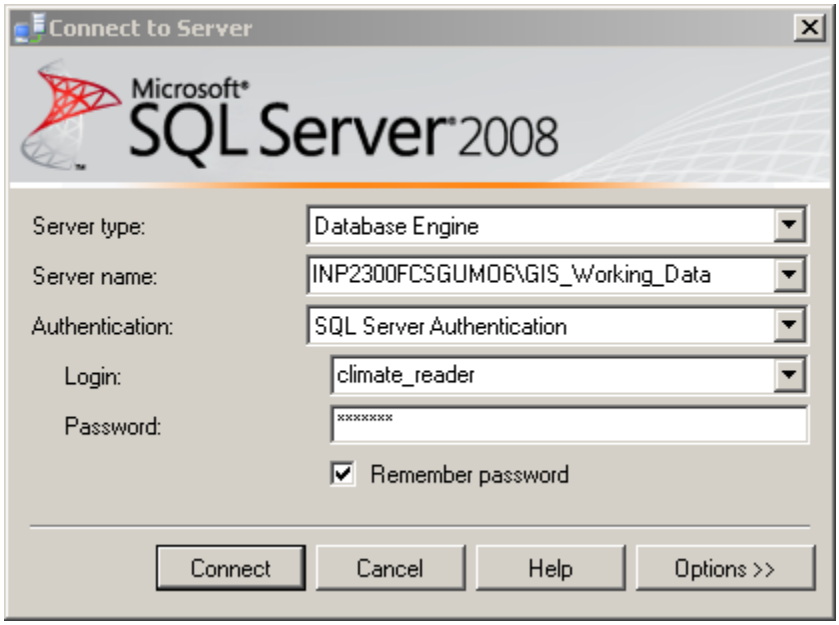
Windows XP or Windows 7 download:
http://www.microsoft.com/downloads/en/details.aspx?familyid=08E52AC2-1D62-45F6-9A4A-4B76A8564A2B&displaylang=en

SQL Server Express home page:
http://www.microsoft.com/express/Database/InstallOptions.aspx

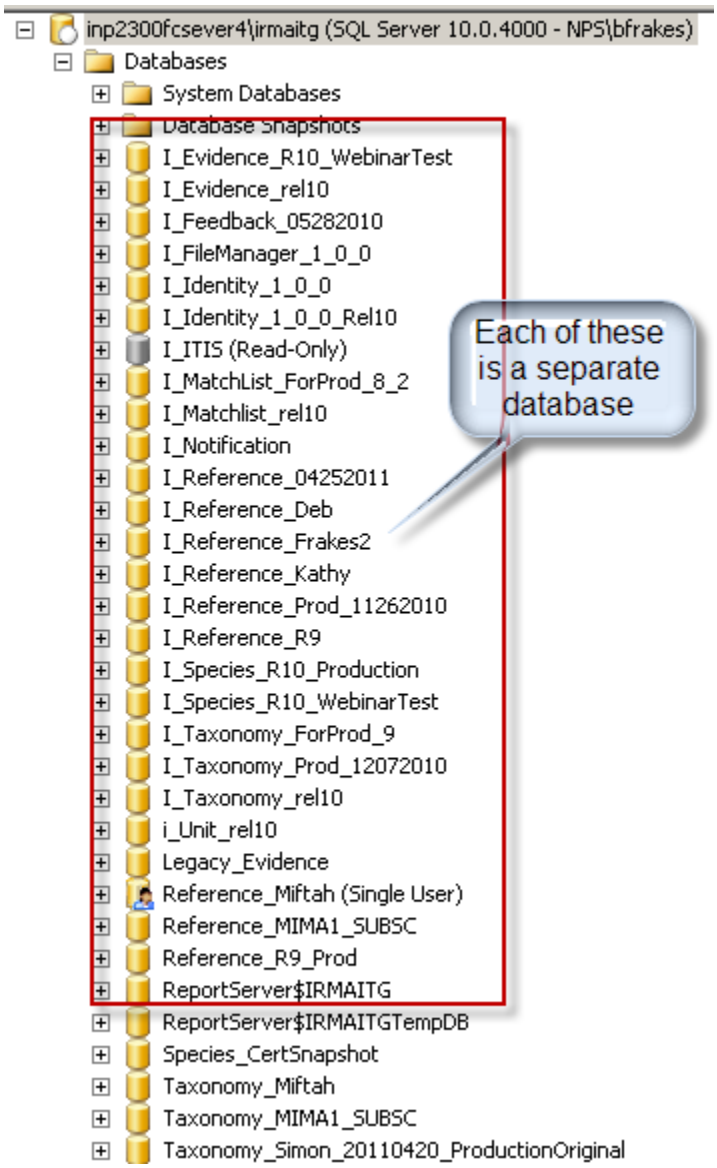
The 64-bit **Management Tools** package (163MB) installed and connected successfully on a Windows 7 64-bit computer when executed as administrator.

NOTE: The following screen shots may look different depending on your installed version of SSME and the database you are connecting to.

2. Run SQL Server Management Studio Express and 'Connect to the Server':



3. Select the appropriate database from the set of databases:



6.2. Python

There are a number of libraries that can be used to connect to SQL-Server. We have found the pyodbc works best and is well documented at (http://code.google.com/p/pyodbc/):

1. Download pyodbc from the following location: http://code.google.com/p/pyodbc/downloads/list. Be sure to get the correct version for your version of python.

2. Run the following example python script. Note that you will need to change the following (highlighted in yellow):

    - Server – The location of the SQL Server
    - UID – The User ID
    - PWD – the Password
    - strSQL – You will need to define a SQL string which is appropriate for the database you are using.

```
########################################################################
##Created by Brent Frakes 6/2/2011
##This demo script provides an example of how to connect to a SQL-Server Database
########################################################
#import pyodbc library
Import pyodbc

#Establish connection to SQL-Server 2008 Database using SQL-Server Authentication
cnxn = pyodbc.connect('DRIVER={SQL Server Native Client
10.0};SERVER=inp2300fcsdeto6\data2008;UID=climate_reader;PWD=cl1m@te')

#Establish connection to SQL-Server 2008 Database using DSN and Windows Authentication
cnxn = pyodbc.connect(';DSN = ClimateDB;Trusted_Connection =yes')


#Create a cursor which manages attributes of data returned from the SQL-Server data source
cursor = cnxn.cursor()

#define SQL Query
strSQL = "SELECT TOP(100)* FROM viewer.vBasicSelection WHERE VersionID=1'"

#Execute SQL Query
cursor.execute(strSQL)
a = cursor.fetchall()

#print results to screen
print a

#close the connection
cursor.close()
cnxn.close()

########################################################################
```

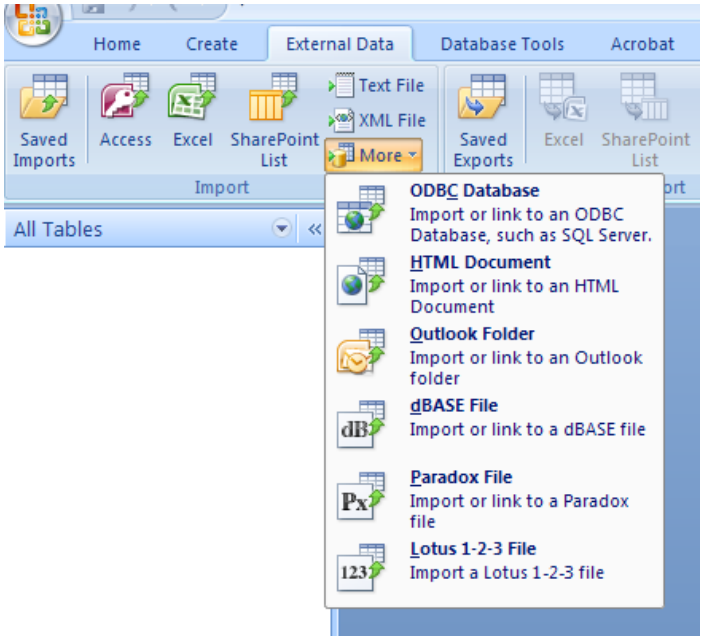
Python returns the data as a tuple.

6.3. MS Access

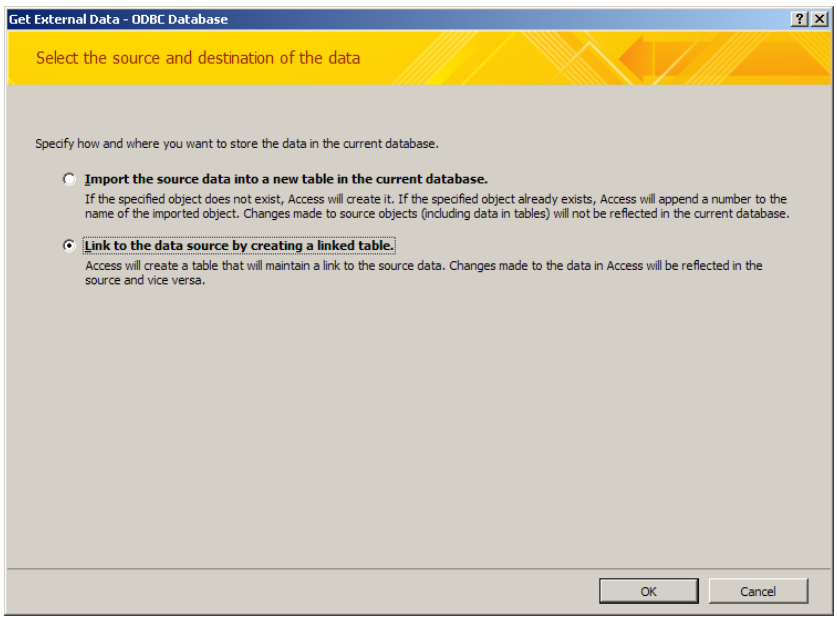
This example utilizes an existing ODBC connection.

1. Start Access and create a new database

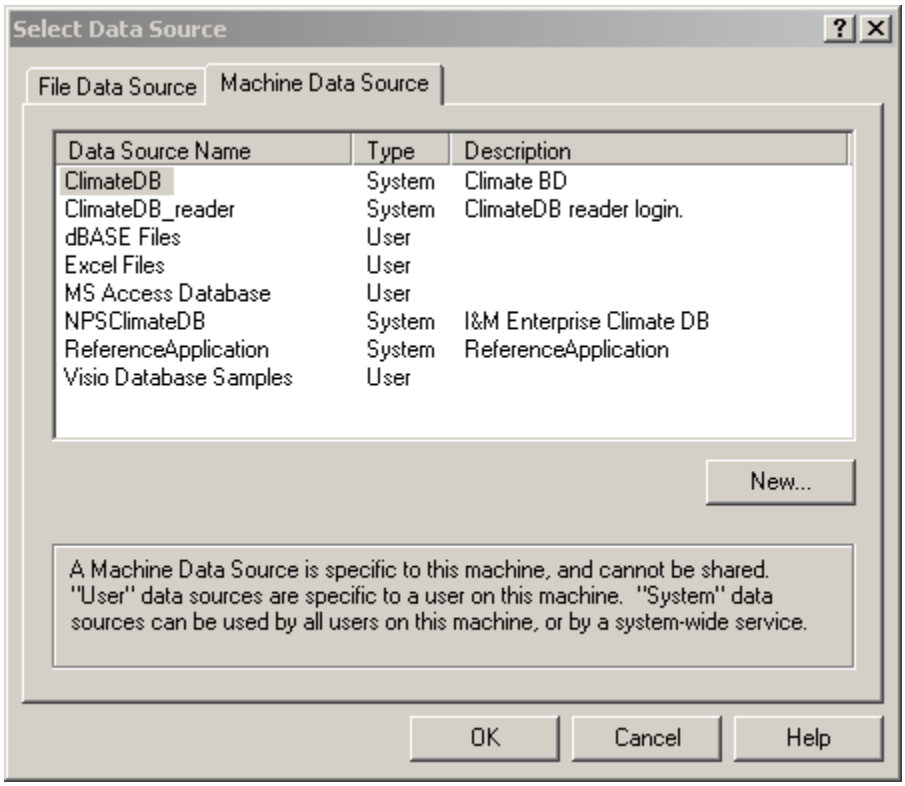
2. Select External Data-->More-->ODBC Database



3. Select the option to Link the Data Source

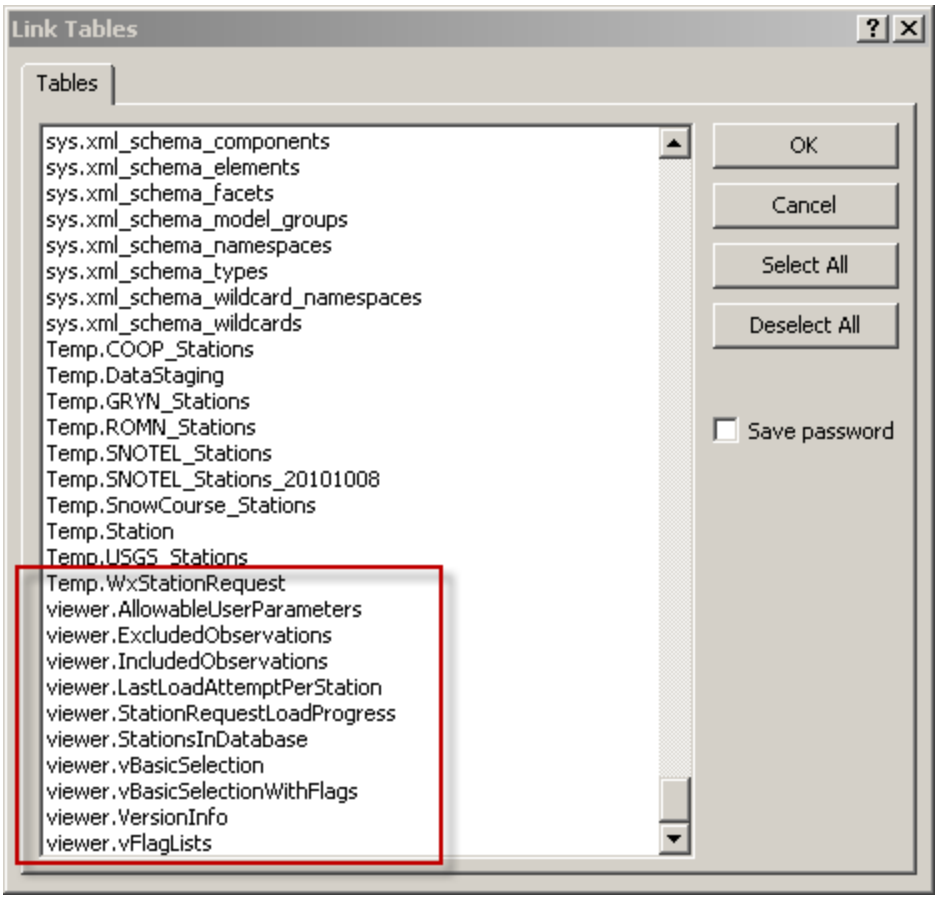


4. Select the Machine Data Source tab and find the appropriate data source. Hit OK.



5. Select the Appropriate Objects

You can select tables or views (virtual tables).

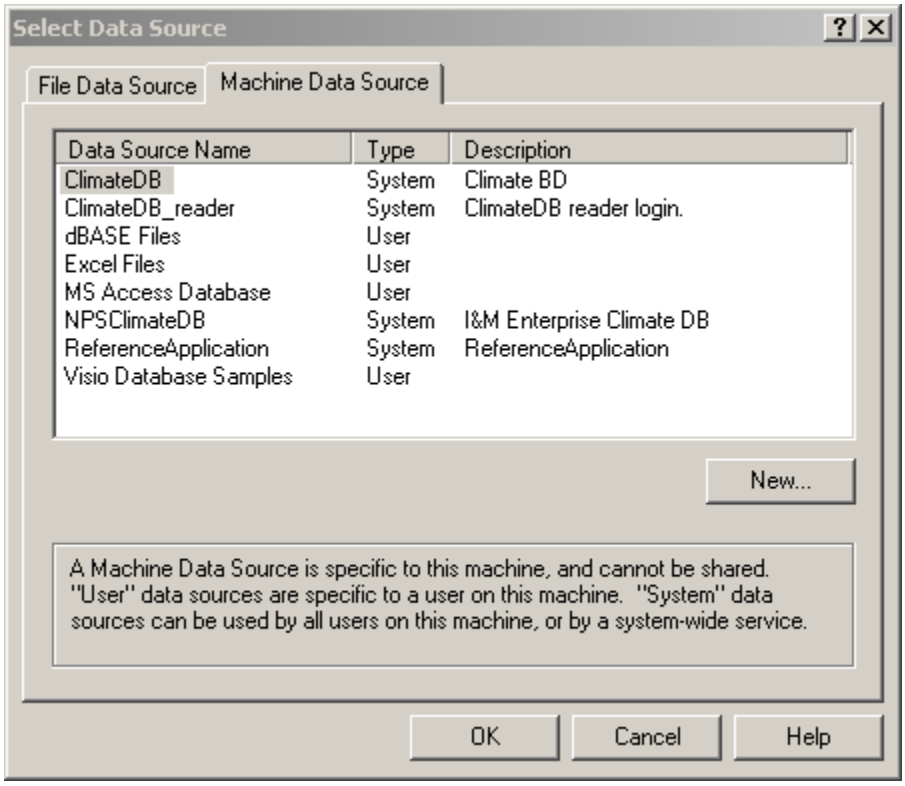


While MS Access works best with tables and views, it is still possible to execute Table- and Scalar-Valued Functions from Access by creating a pass-through query. To create a pass-through query, follow these steps:

1. Select Query Design from the Create tab in Access.
2. Close the Show Table window.
3. Select Pass-Through from the Query Type section on the Design tab.

4. Type in your T-SQL statement and click the Run button on the Results section on the Design tab.

5. When prompted on the Select Data Source window, select the Machine Data Source tab and find the data source. Hit ‘OK’.



6. On the SQL Server Login window, check the Use Trusted Connection checkbox and click ‘OK’.

The statement should execute and return results. You can save the query in Access.

6.4. R

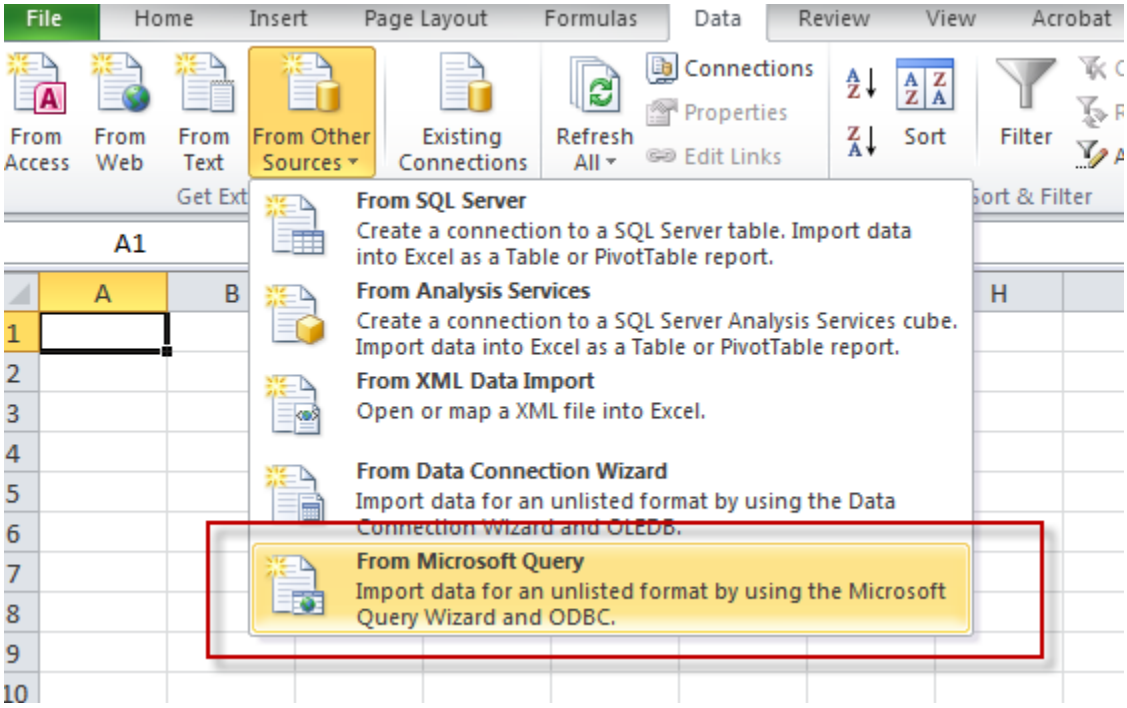
Below is example of how to connect using R where the ODBC data source is called NPS_Climate:

```
      # Only 32-bit R!!!
library(RODBC) #load the RODBC library into your R session, each time you use RODBC function
climate.db <- odbcConnect("NPS_climate",uid="climate_reader",pwd="*******")  #**** is whatever password you were given
sqlTables(climate.db)
# stick SQL queries from the SOP into character variables, paste() used to allow multiple lines in code
q1 <- paste("SELECT su.UnitCode,sid.*",
      "FROM [viewer].[StationCDBCodeUnitCode] su ",
      "INNER JOIN viewer.StationsInDatabase sid ",
      "ON su.CDB_Code = sid.CDB_Code ",
      "WHERE UnitCode IN ('GRTE','YELL') ",sep="")
StationList <- sqlQuery(climate.db,q1)
# Insert other commands here
close(climate.db)
```

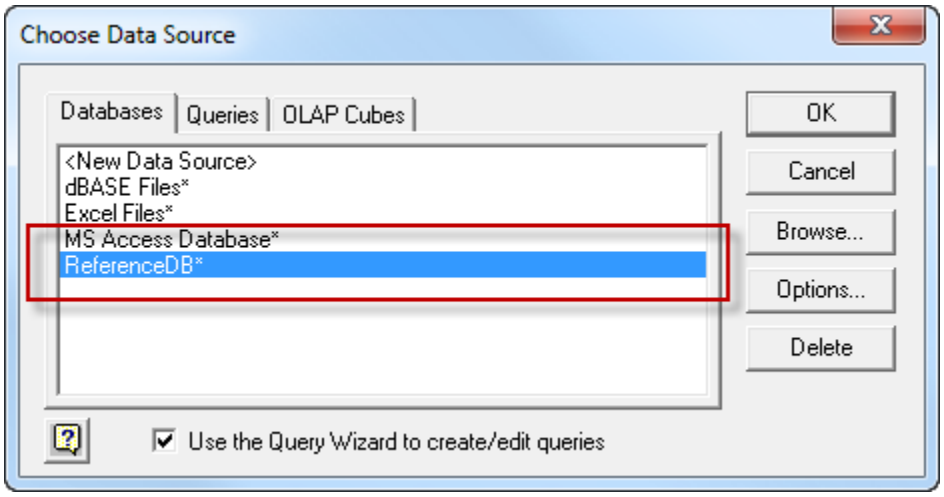
6.5. MS Excel

This example utilizes an existing ODBC connection.

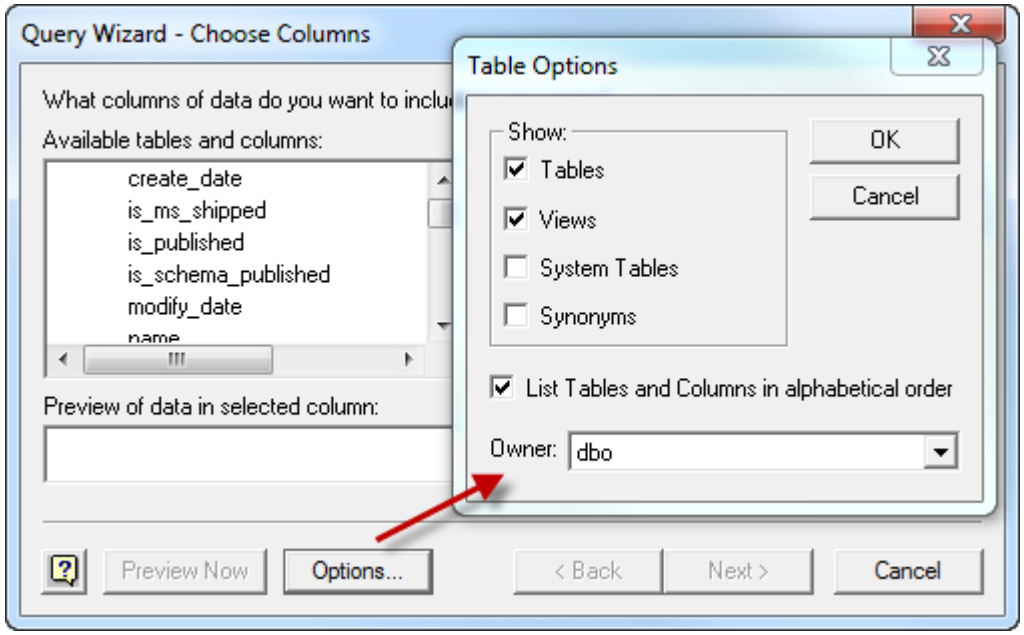
1. Select Data-->From Other Sources-->From Microsoft Query



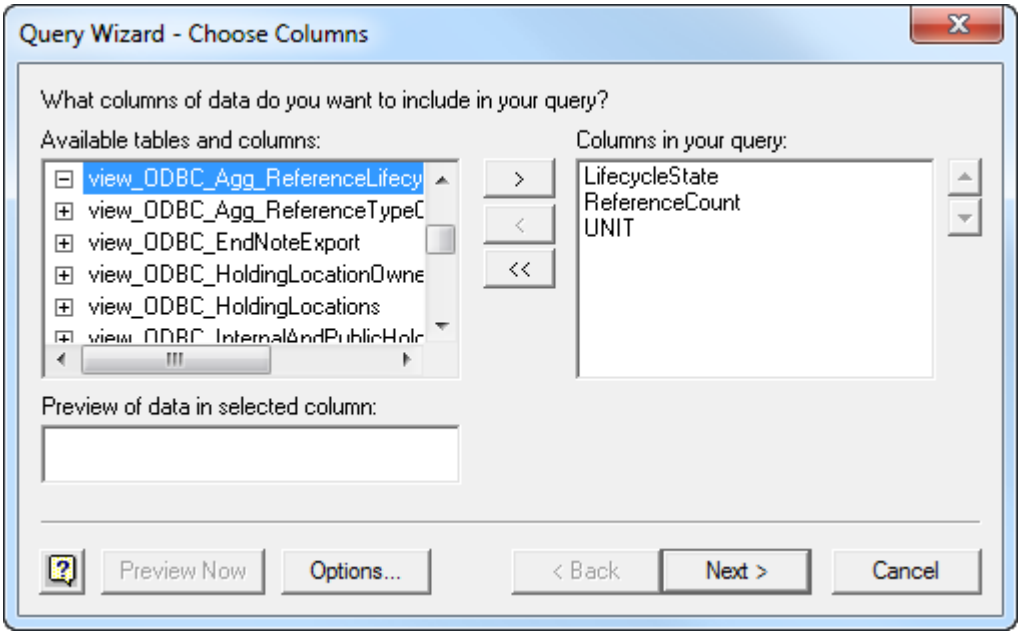
2. Select the appropriate Data Source (Example here is for ReferenceDB). Make sure the “Use Query Wizard to Create/Edit Queries” is checked.



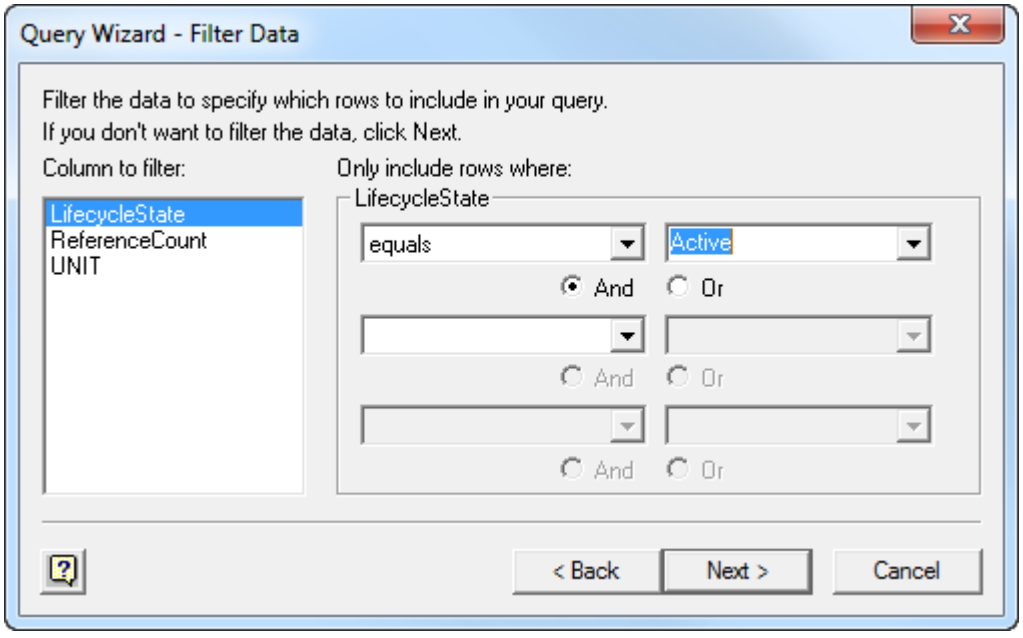
3. Select the appropriate tables or views. If you don’t see them, select options and specify the appropriate types of information to show.



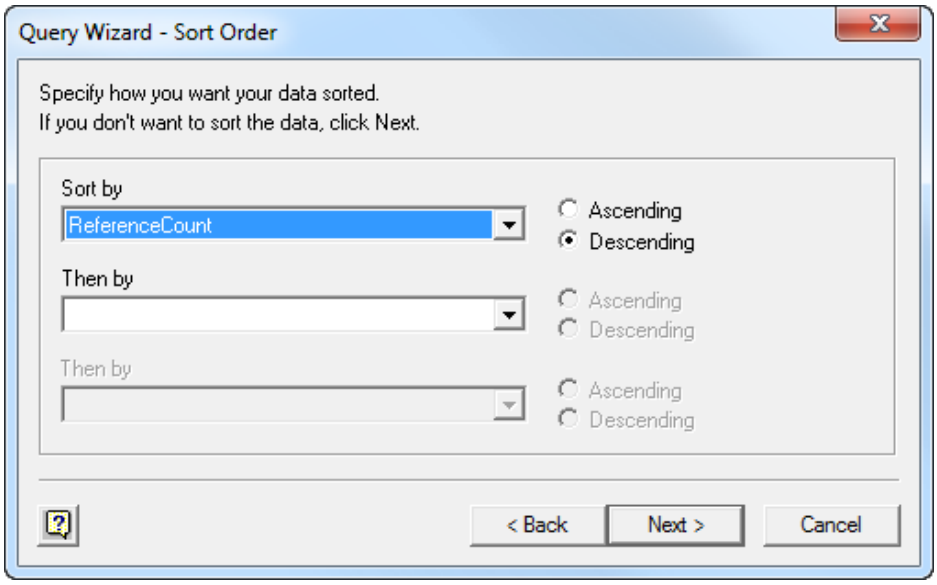
4. Select Data to Include in Query



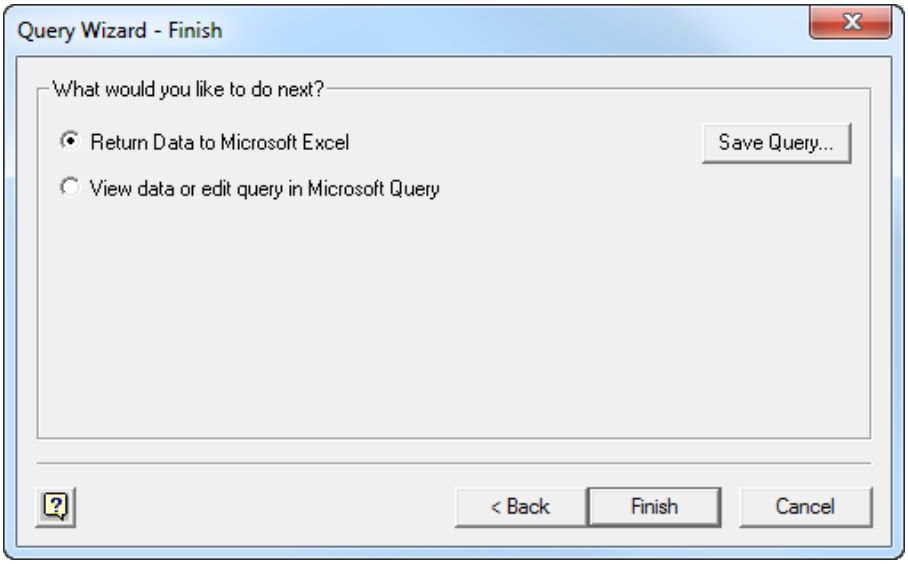
5. Optionally Specify Any Filter Criteria



6. Optionally Specify Sort Order



7. Run the query by selecting “Return Data to Microsoft Excel” or get more advanced query options by selecting “View data or edit query in Microsoft Query”

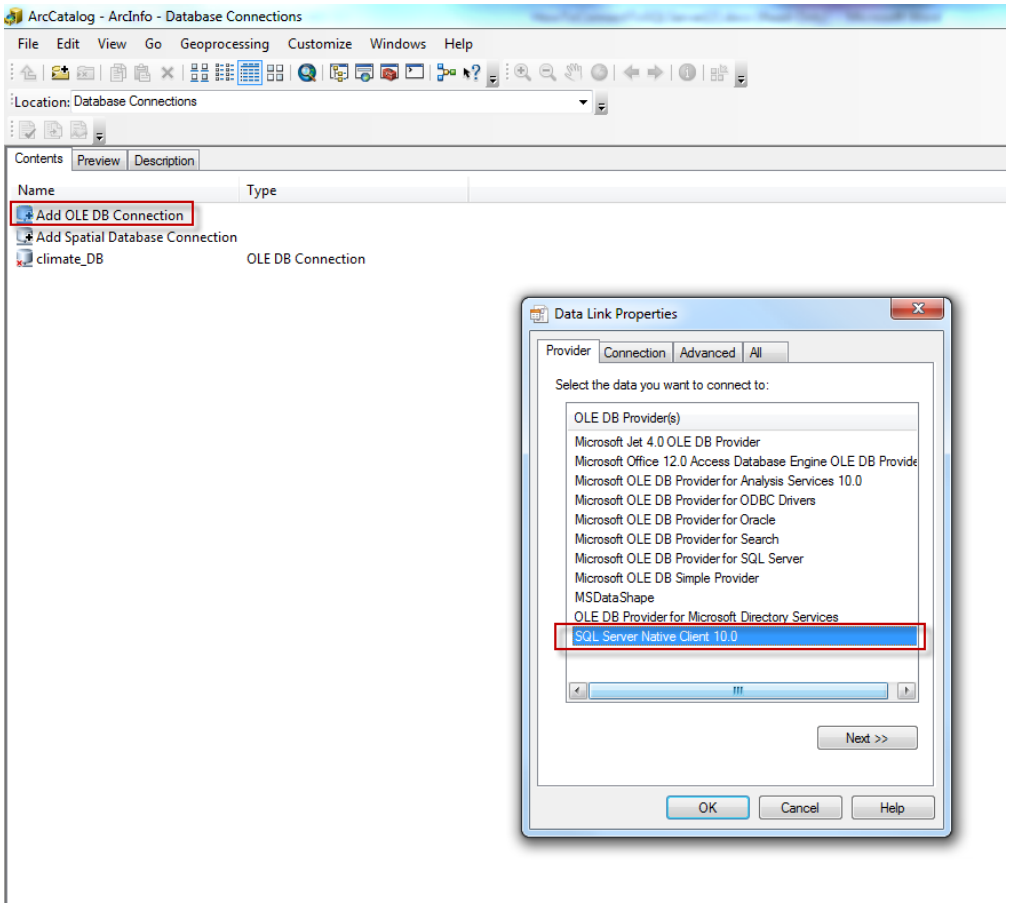


6.6. ArcGIS 10

Process

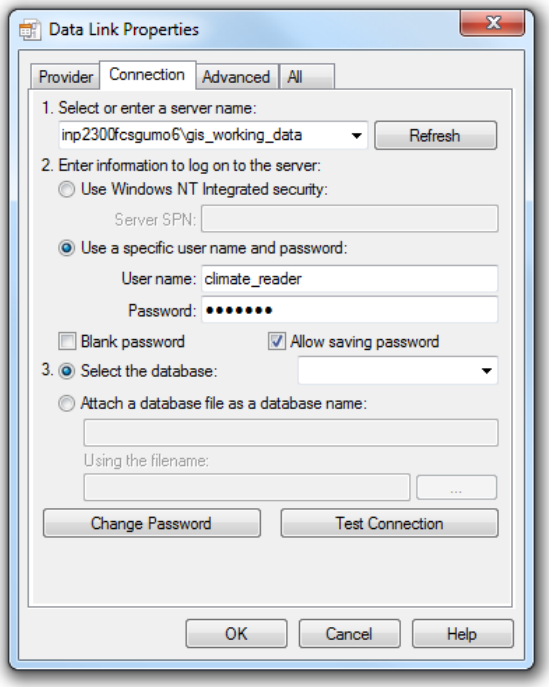
1. Start ArcCatalog

2. Click ‘Add OLE DB Connection’ -> Data Link Properties -> Provider tab->SQL Server Native Client 10.0->Click Next.



3. Fill out the Database Connection Properties dialog box as shown below:

- Enter the server name
- Enter the User Name
- Unclick ‘Blank password’ and enter Password
- Click ‘Allow saving password’
- Click ‘Test Connection’. If successful, then press OK.



Provide a name for the database connection. Type any name you wish, then press enter.

4. Open ArcMap

5. Add Data->Database Connections